

**A Successful Example of a Layered Architecture Based
Embedded
Development with Ada 83 for Standard-Missile
Control**

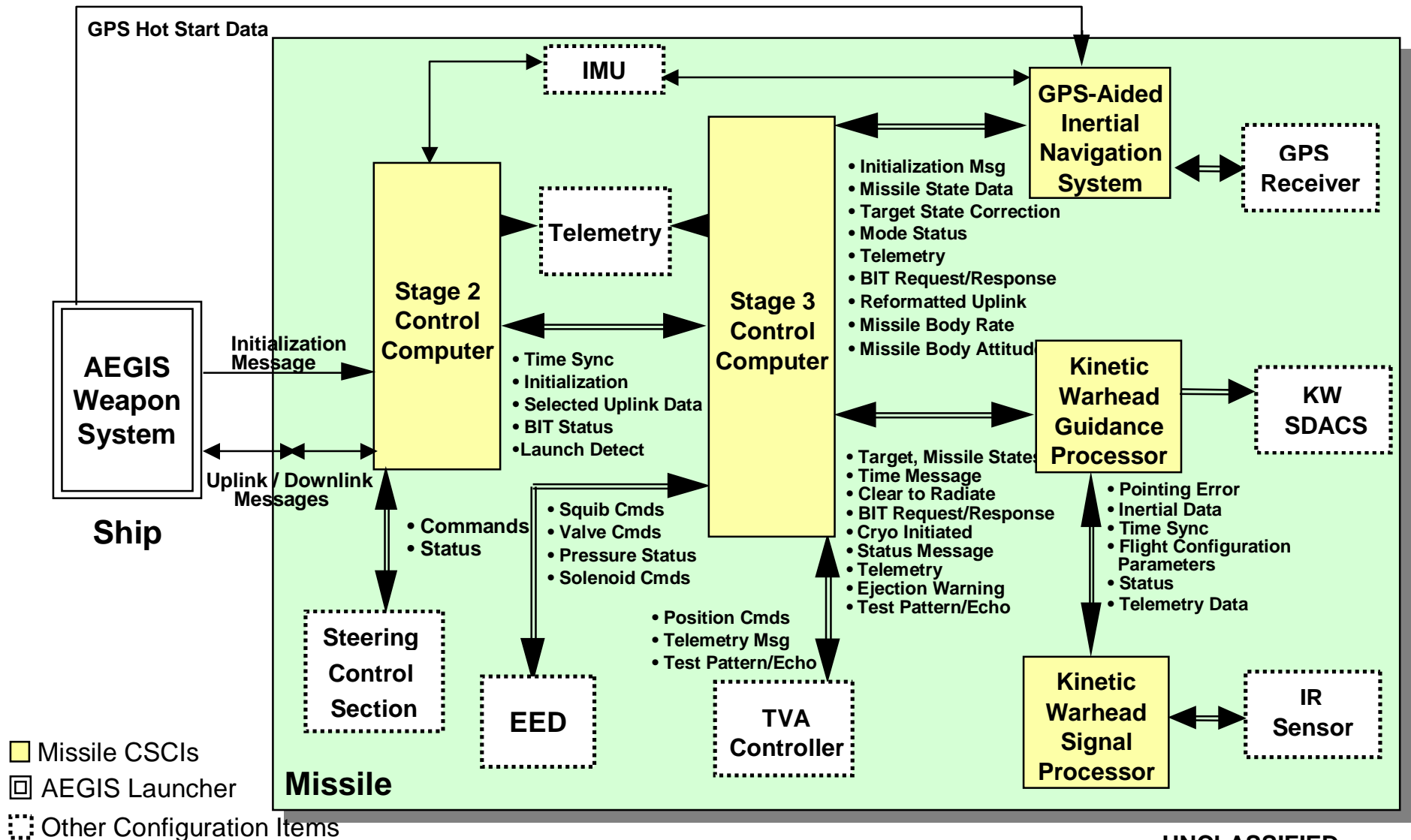
**Kelly L. Spicer
Raytheon Missile Systems
Missile Software Engineering Center
Tucson, Arizona
520-663-7020
klspicer@west.raytheon.com**

Nov 14, 2000

Overview

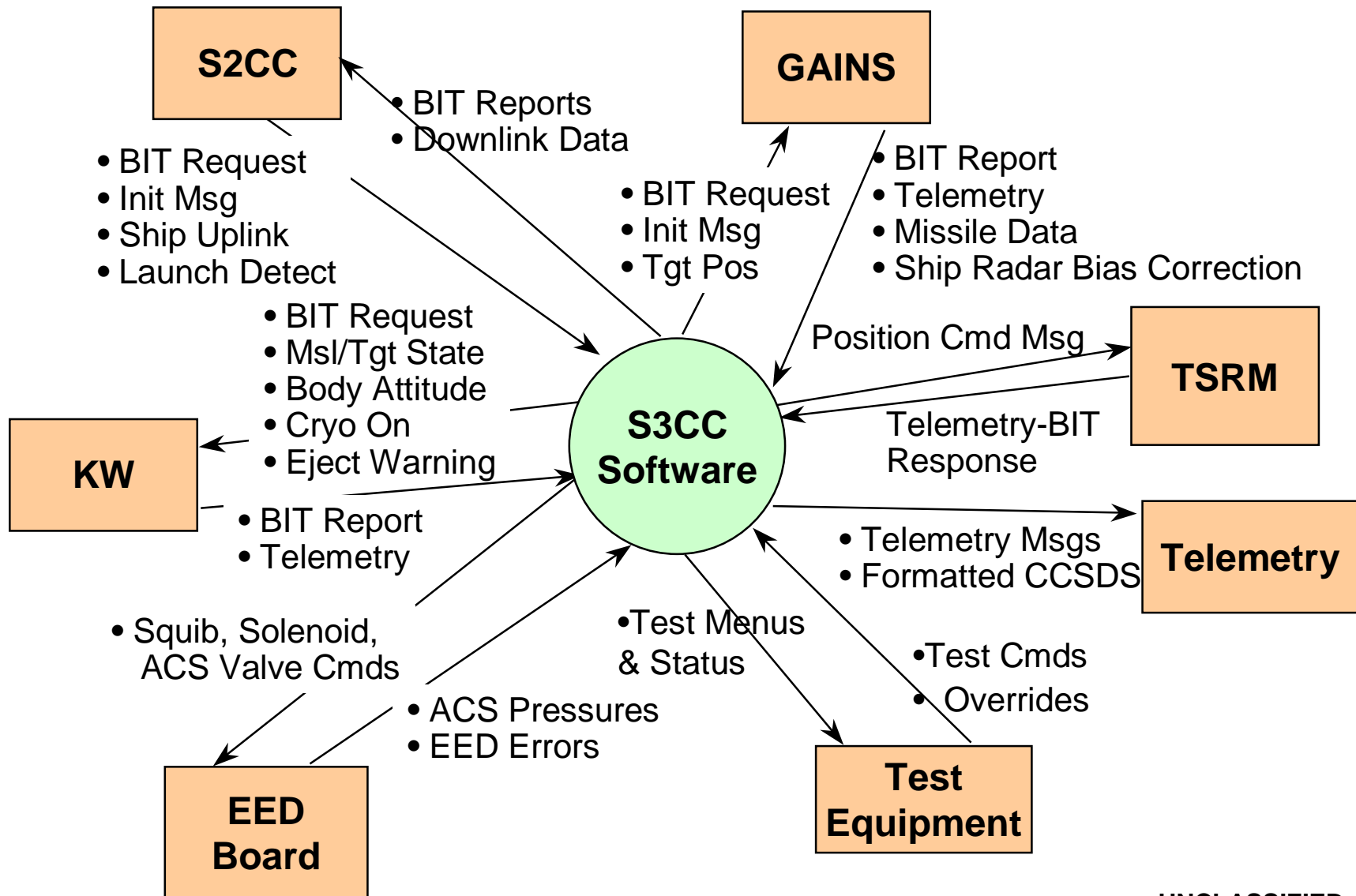
- **SM-3 Software Overview**
- **S3CC Context Diagram**
- **S3CC Requirements**
- **Design Notation**
- **Software Architecture Overview**
- **Layering Scheme Description**
 - Purpose of Each Layer
 - Conventions for Each Layer
 - Examples
- **Build Approach Using Layers**
- **A little about Tasking and the Block IV Kernel**
- **Reuse of Algorithms**
- **Reuse - Architecture Typing (Archi-typing)**

SM-3 CSCIs



UNCLASSIFIED

S3CC Software Context Diagram



S3CC SW Requirements Summary

- **Interfaces**

- Serial (TSRM, GAINS, KW, Telemetry)
- Direct control (ACS, squibs, solenoids, discretetes)
- MLI bus (stage 2)

- **Mission state control, two timelines:**

- Since launch,
- To-go (to prepare and eject KW)

- **BIT - functions and reporting**

- **Autopilot/attitude control (100 Hz updates), three modes:**

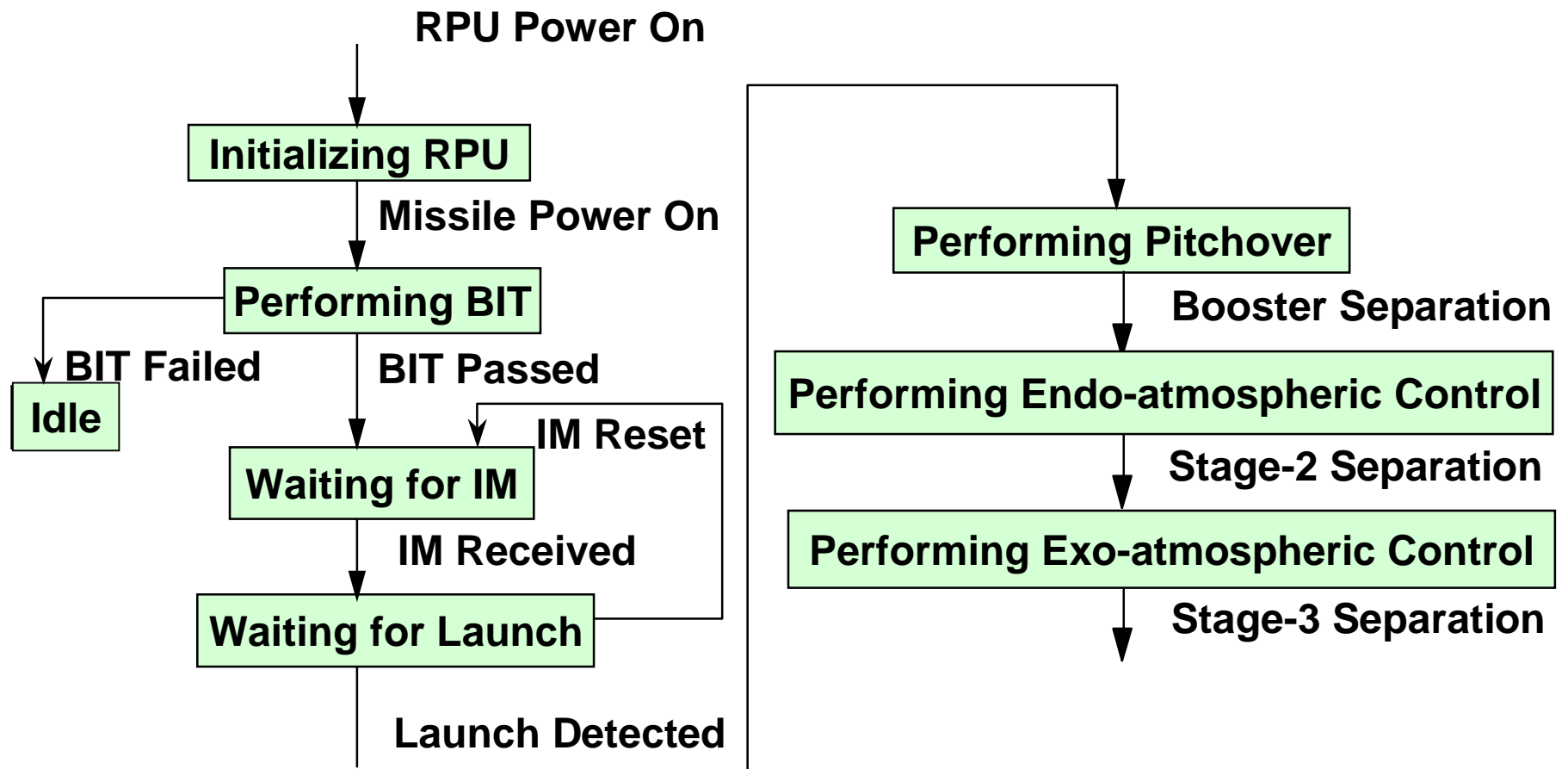
- Cold gas, warm gas, Thrust Vector Control (TVC)

- **Guidance (10 Hz updates)**

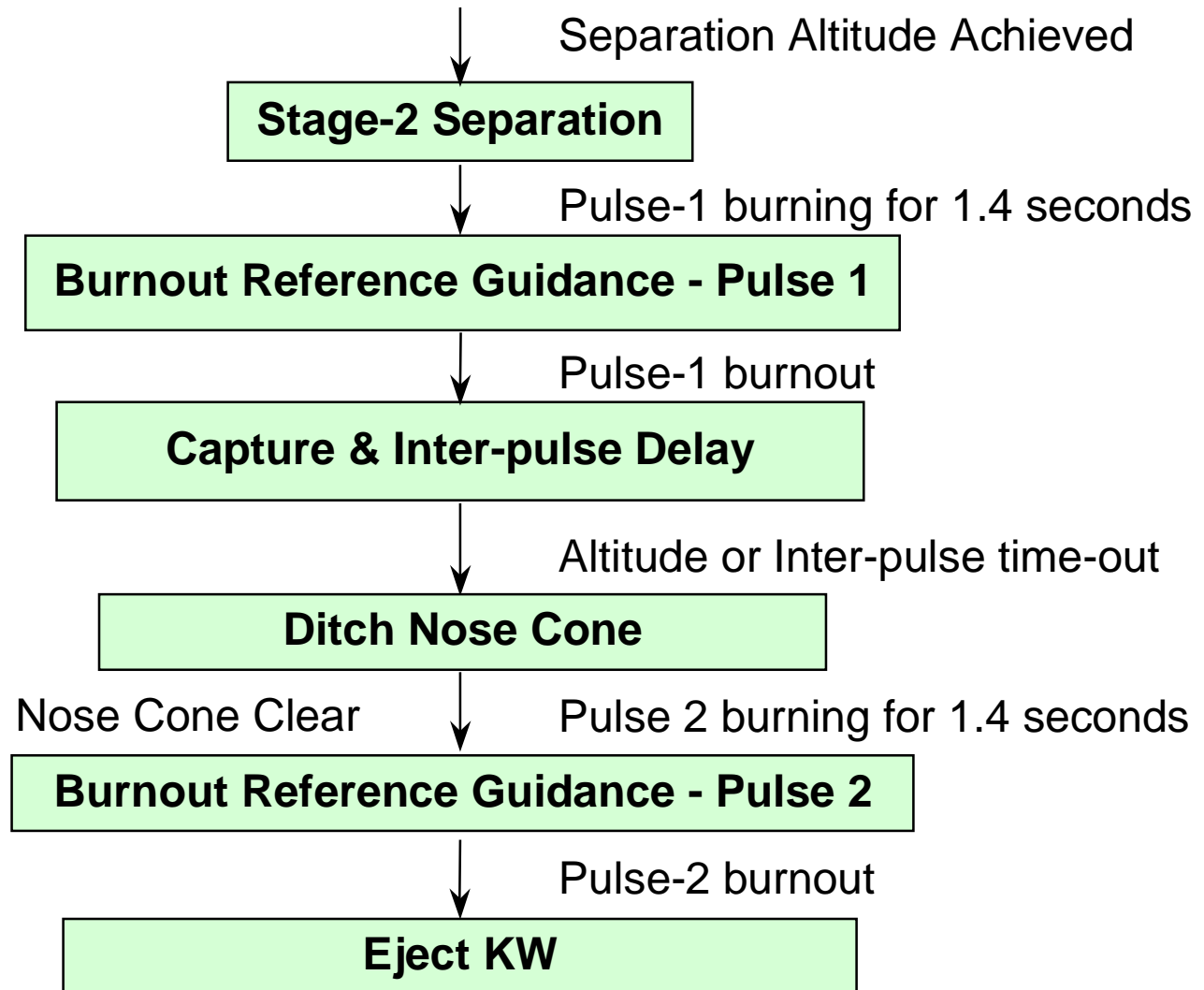
- Burnout reference guidance
- Adjust for non-nominal TSRM burn-rate

- **Adaptation parameters (capability to load system-defined constants separate from software)**

System-Level State Transition Diagram

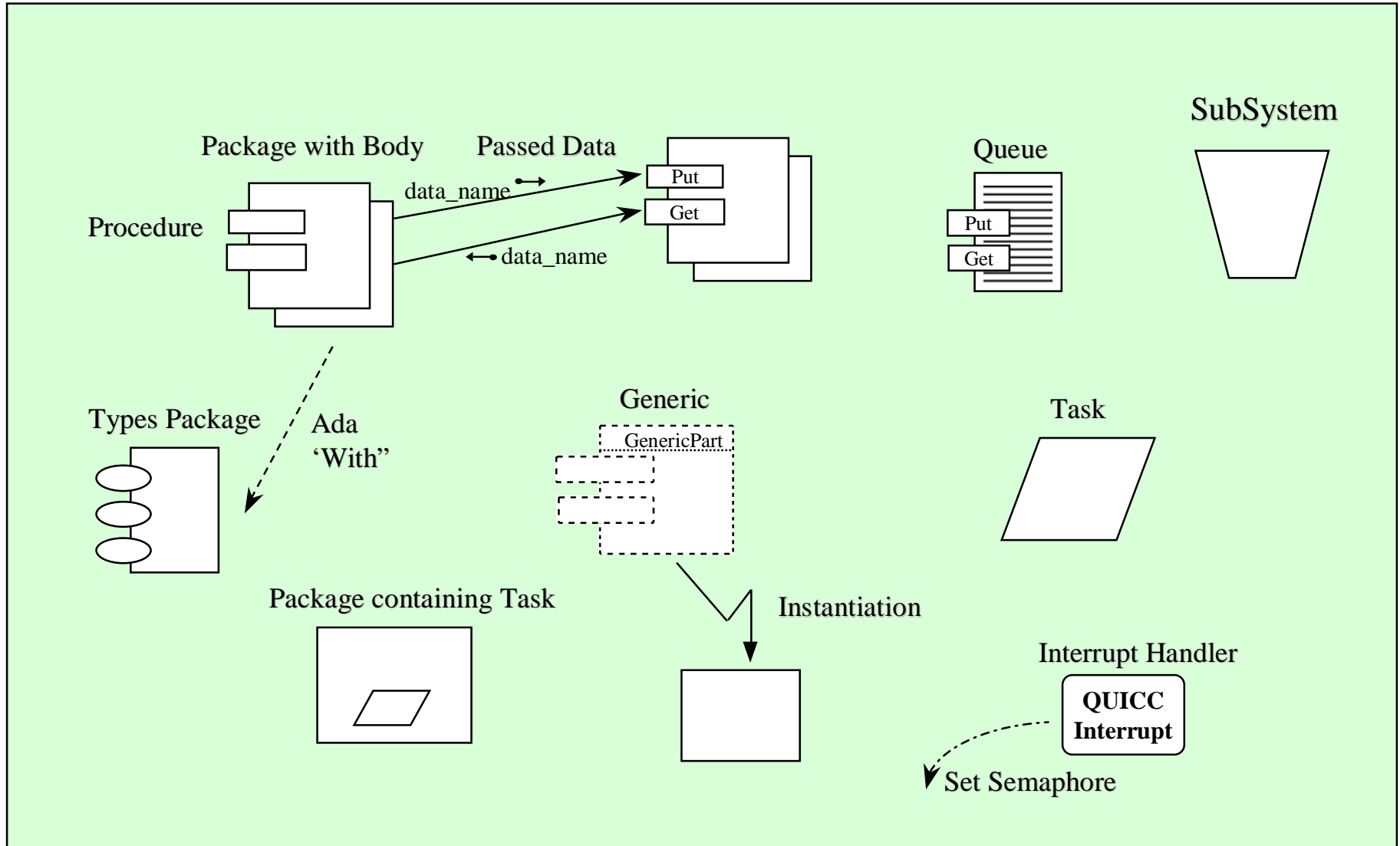


S3CC Mission Sequencer States

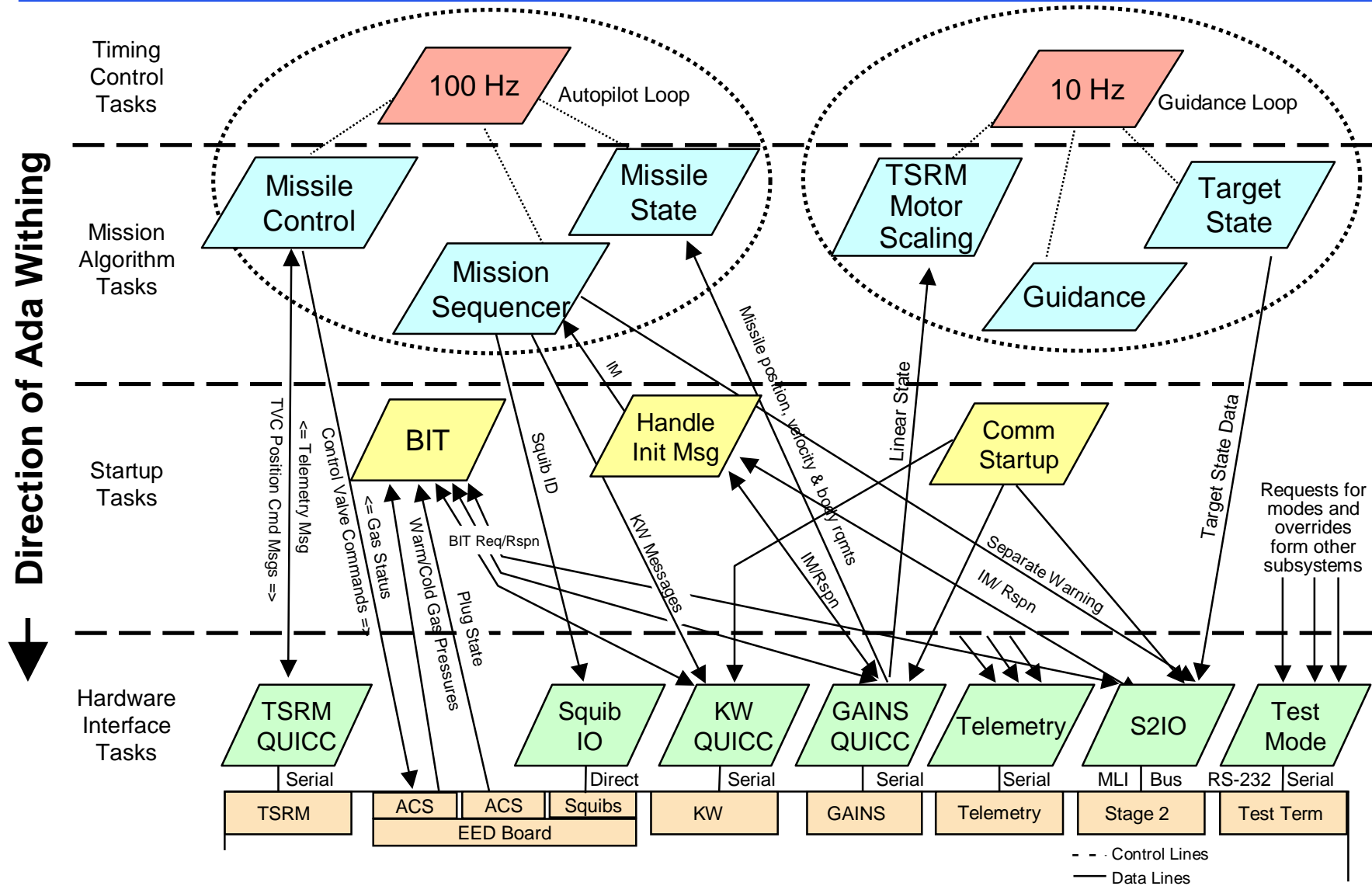


UNCLASSIFIED

Buhr-Based Design Notation

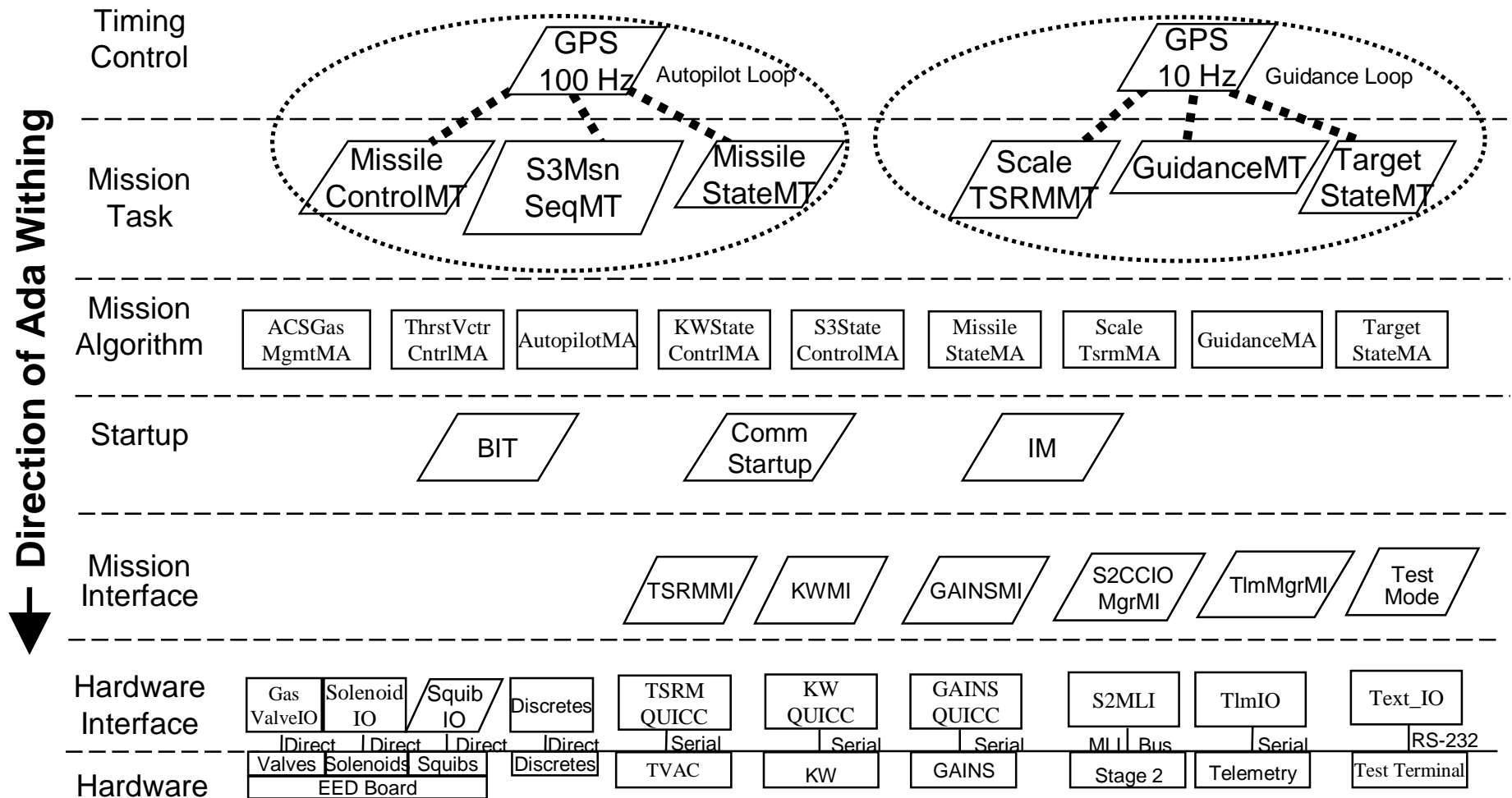


S3CC Task Hierarchy



UNCLASSIFIED

SM-3 Stage-3 Software Architecture



UNCLASSIFIED

Stage-3 Layering Scheme

- Hierarchical Approach
- Lower Layers Provide Services
- Higher Layers Provide Management
- Commands Flow Down
- Data Flows up and Down
- No “withing” up the Architecture, Down Only

6 Logical Layers in the Hierarchy

6. Timing Control

Mission

5. Mission Task

4. Mission Algorithm

3. Startup

2. Mission Interface

1. Hardware Interface

1. Hardware Interface

- Hardware Direct Interface
- Works in Hardware Addresses, Register Formats, Interrupt Handling, etc.
- Generics Used Across Interfaces when they're Sufficiently Similar

2. Mission Interface

- Provides a Consistent, High-Level I/O Interface to the Mission Tasks
- Implements Interface Protocols defined in the IRSs
- Contains Tasks for Completion of Receiving Message Data on External Interface.
- Converts Data Between External and Internal Format
- Transfers Data to/From Store Manager

3. Startup

- Calls the Mission Interface to Execute Startup Activities Like Starting the Interfaces, Conducting BIT, & Handling the Initialization Message
- Startup Tasks Terminate Themselves Before Launch After Their Job is Done

4. & 5 Mission

- The Mission Layer Implements Most of the Mission Algorithms and Logic Defined as Requirements in the SRS
- Implements All Mission Intelligence (i.e. Guidance, Autopilot, Missile State Control, etc.)
- Divided into two sub-layers: Mission-Task and Mission-Algorithm
 - Tasks gate algorithms at proper/coordinated periodic rate
- Portable Because it runs on the Virtual Machine provided by the Mission-Interface Layer below (and some Blk IV Kernel calls for time)

6. Timing Control

- Provides Timing Control for the Mission-Level Tasks

Layer Naming Conventions

- Mission-Task Layer : “MT” Suffix
- Mission-Algorithm Layer : “MA” Suffix
- Mission-Interface Layer : “MI” Suffix
- Subsystem mnemonic (e.g. ThrstVctrCntrl)
 - ThrstVctrCntrlMA
 - ThrstVctrCntrlIntfDefs

Layer Interface Conventions: Hardware-Interface

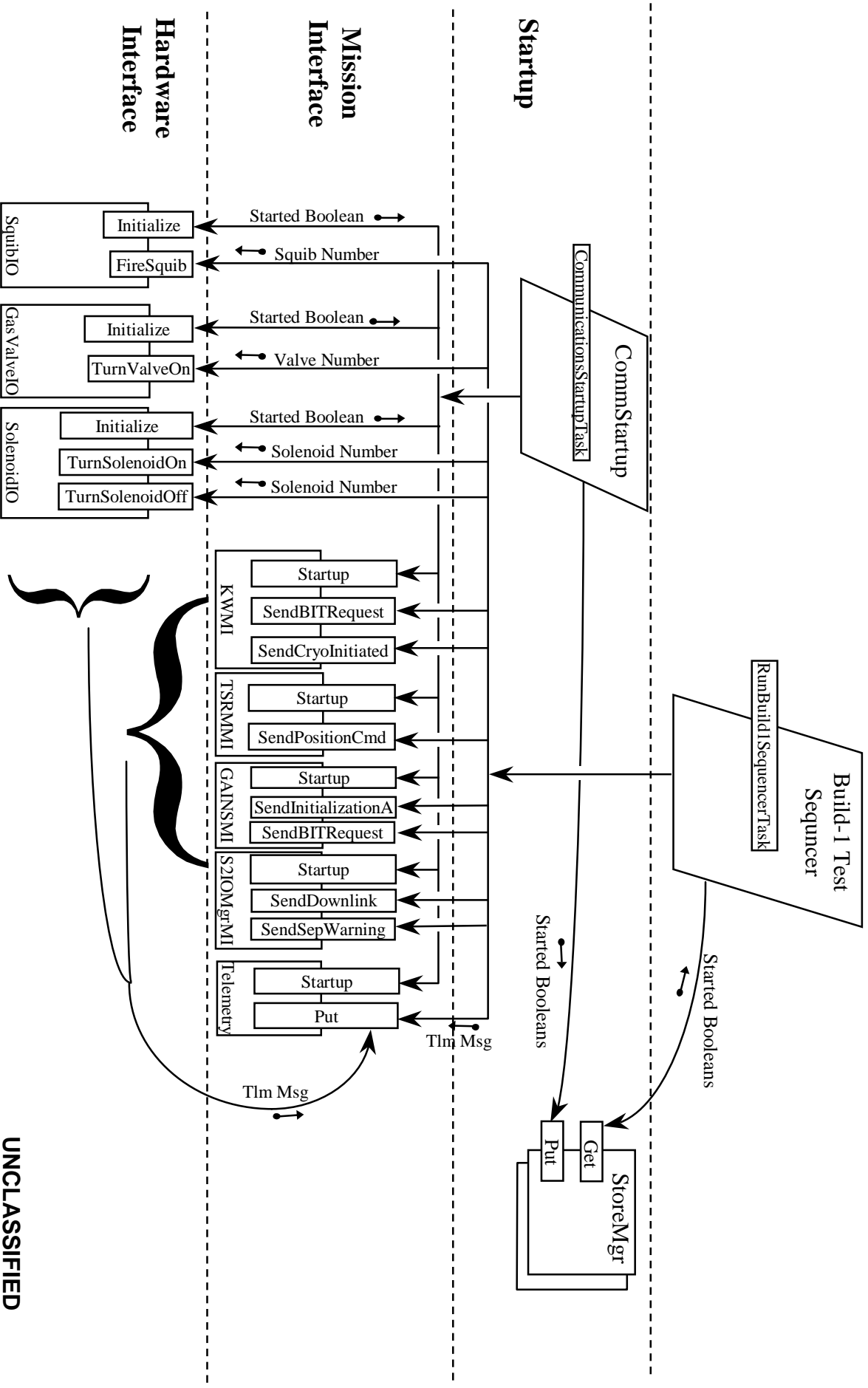
- **Each interface package exports these procedures:**

- Initialize - Called by CommStartup or the “MI” Level
- Functional Procedures to conduct Hardware Operations (i.e. “FireSquib,” “TurnValveOn,” etc.)
- Since There is no Ada “withing” or Procedure calls up the Hierarchy, in-coming data is put into queues, then semaphores are “set” to signal the upper layer(s) that message data has arrived

Layer Interface Conventions: Mission Interface

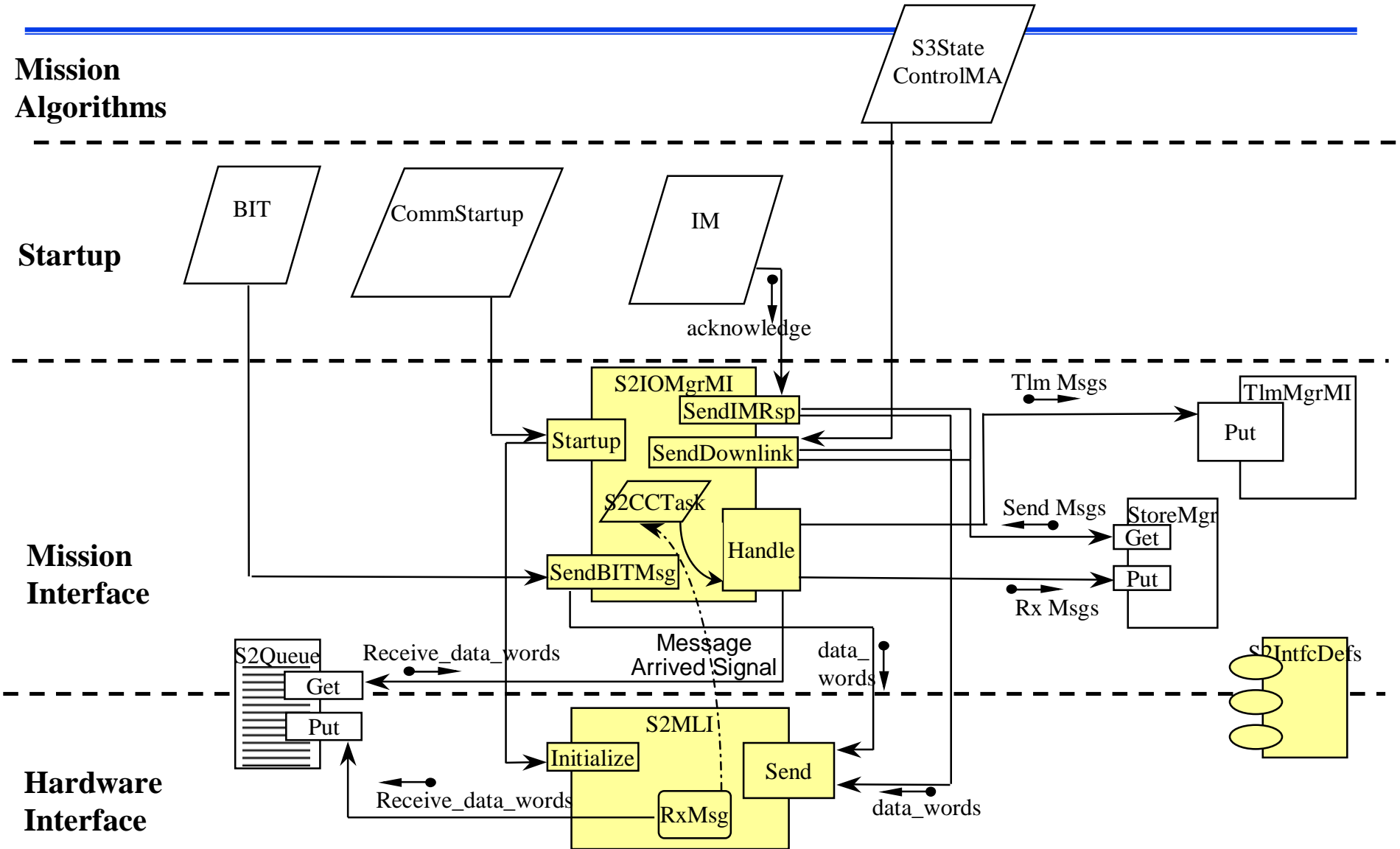
- **Each interface package exports these procedures:**
 - Startup - Called by CommStartup Task
 - Send<MessageName> One for each message able to be sent on the Interface. Parameterless, message data retrieved from StoreMgr
- **Each Interface Subsystem has “InterfaceDefs” Package for defining Internal interface.**
- **Each Interface Subsystem has “MessageDefs” Package for defining External Interface**
 - Rep clause used to match IRS definition bit-for-bit

Example: Build-1 Test Sequencer



UNCLASSIFIED

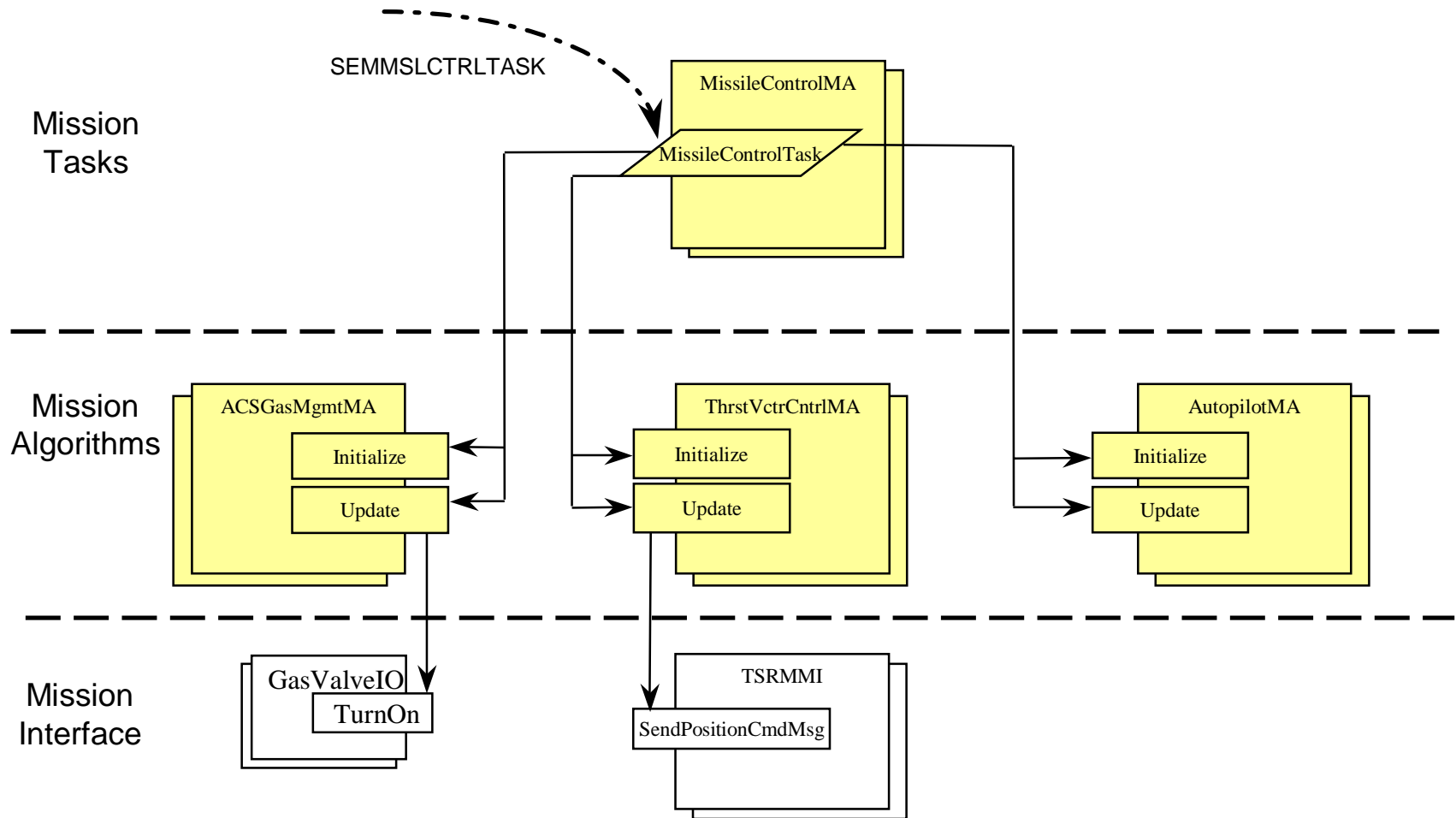
Example: Stage-2 MLI Bus Interface Subsystem



Layer Interface Conventions: Mission-Algorithm Layer

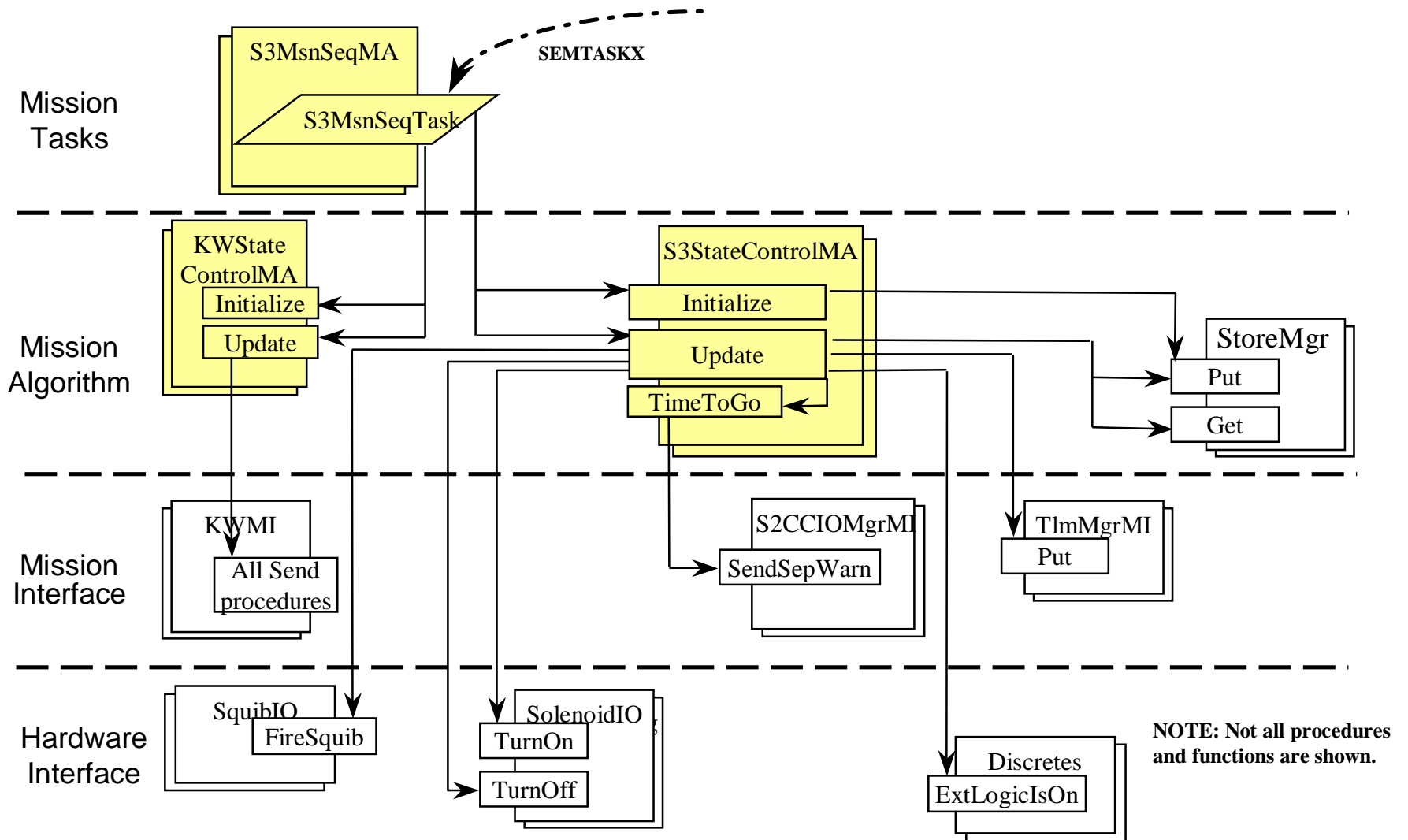
- Each package Exports at least two procedures:
 - “Initialize”
 - “Update”
- Package Bodies
 - In Update Procedure
 - First thing - Storemgr.get(<Package_Name>_Input_Record);
 - Last Thing - Storemgr.put(<Package_Name>_Output_Record);
- InterfaceDefs Packages
 - Should contain definitions for
 - <Package_Name>_Input_Record (input from StoreMgr)
 - <Package_Name>_Output_Record (output to StoreMgr)
 - Types for Instantiating Telemetry Generic (should comply with the data definitions in Telemetry IRS)

Mission-Algorithm/Mission Task Level Example: “Missile Control” Subsystem



UNCLASSIFIED

Mission-Algorithm Level Example: S3 State Control Subsystem



NOTE: Not all procedures and functions are shown.

UNCLASSIFIED

Build Approach Using a Layered Architecture

- 1. First build the Interface Layers, Implementing IRS Requirements**
 - Test the Interface Implementation
- 2. Build the Skeleton for the Rest of the System (Upper Layers)**
 - Verify Task Priorities, etc
- 3. Add the Implementation/Mission Algorithm Details Incrementally in a series of Builds**
 - CSCI Qualification Testing

Build Sequence Leading to Full Flight Capability

-1.0 Interfaces

-1.1 Pre-Launch

-1.2 Software Architecture Skeleton

-1.3 Pre-Separation

-1.4 Missile-Control

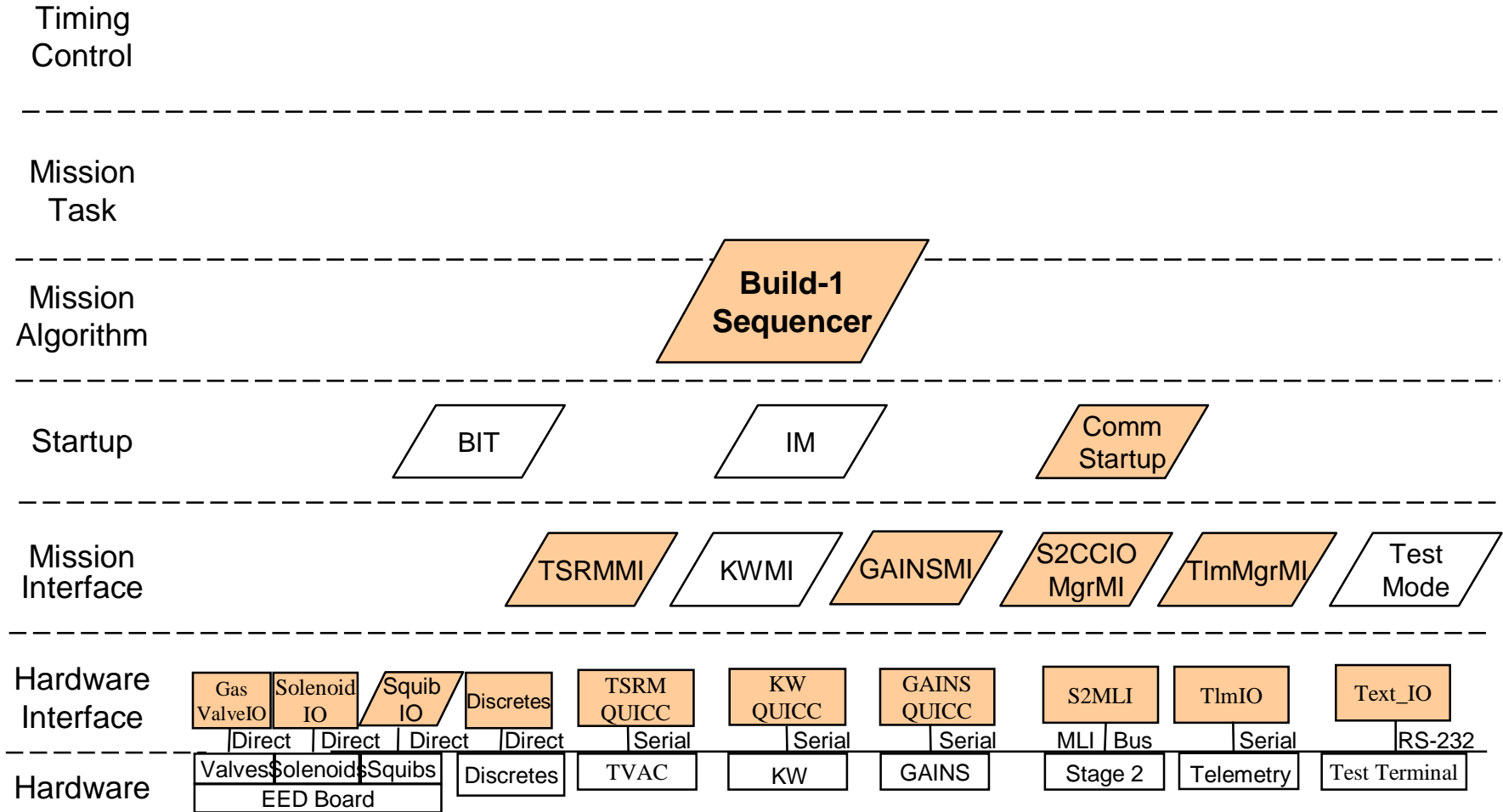
-1.5 Missile-Guidance

-1.6 KW Preparation and Release

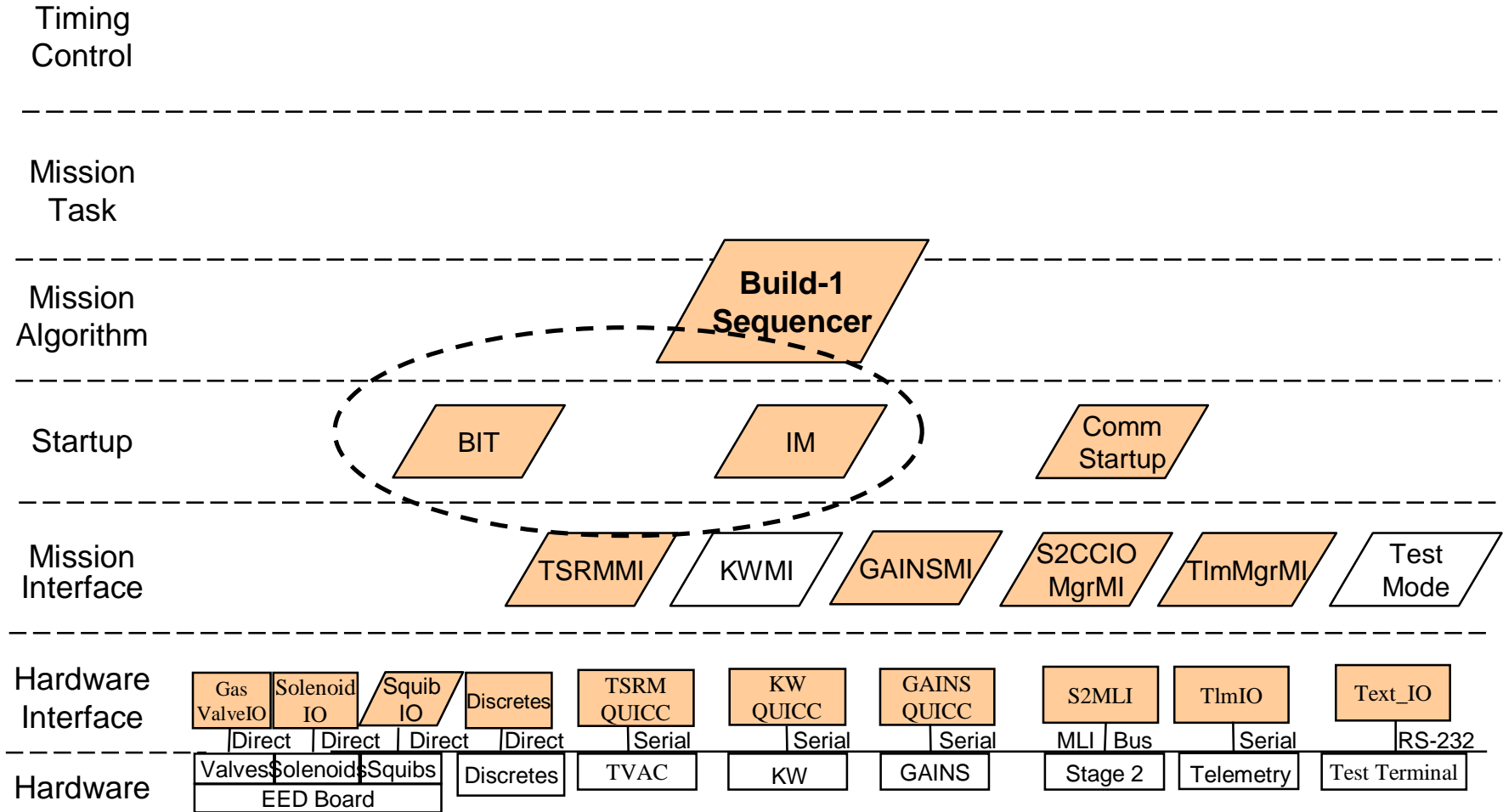
Implement Functions/Algorithms in to Fit Test Schedules

-Order Needed for Flight in this case

Build 1.0 - Interfaces

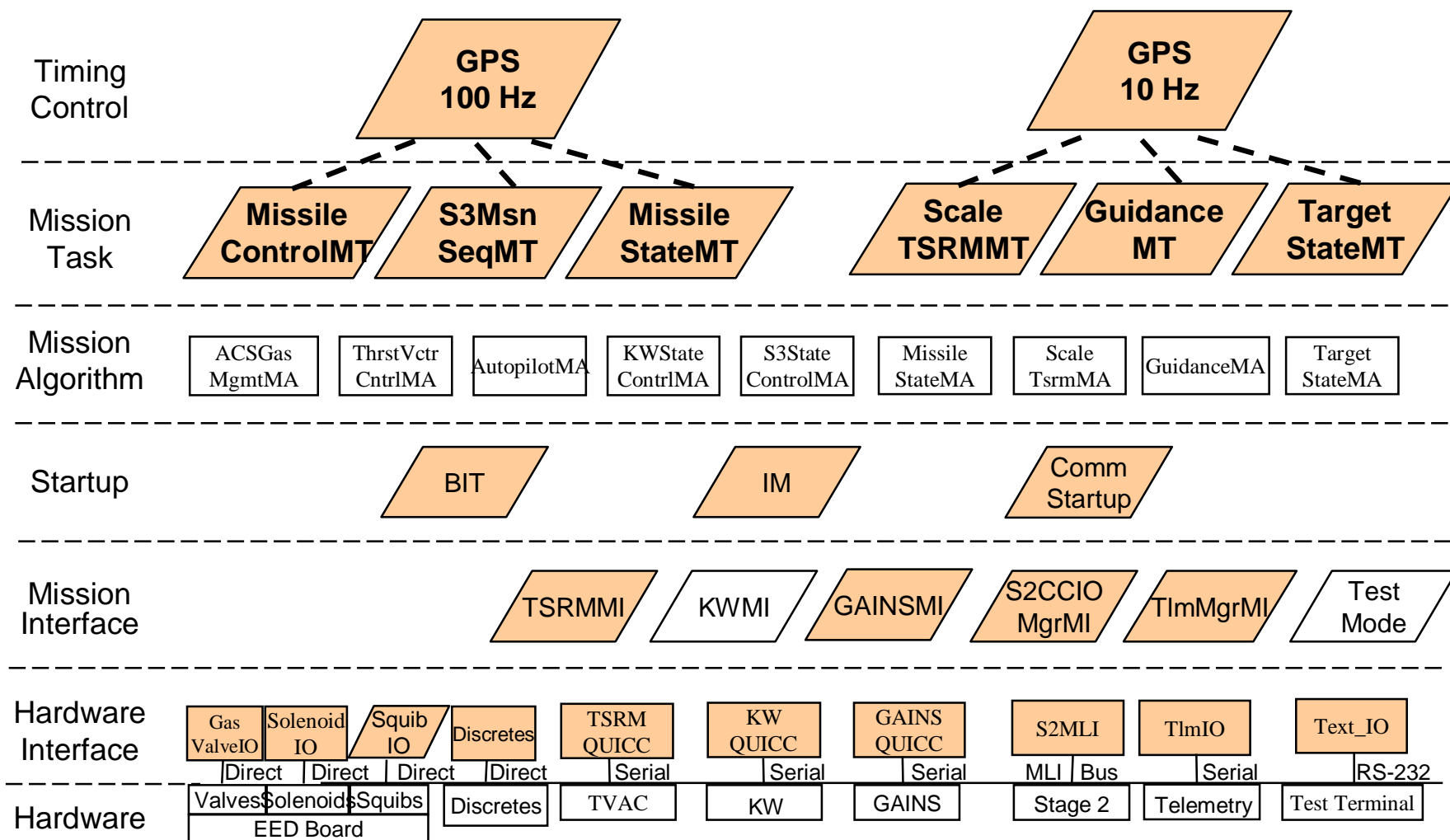


Build 1.1 - Pre-Launch

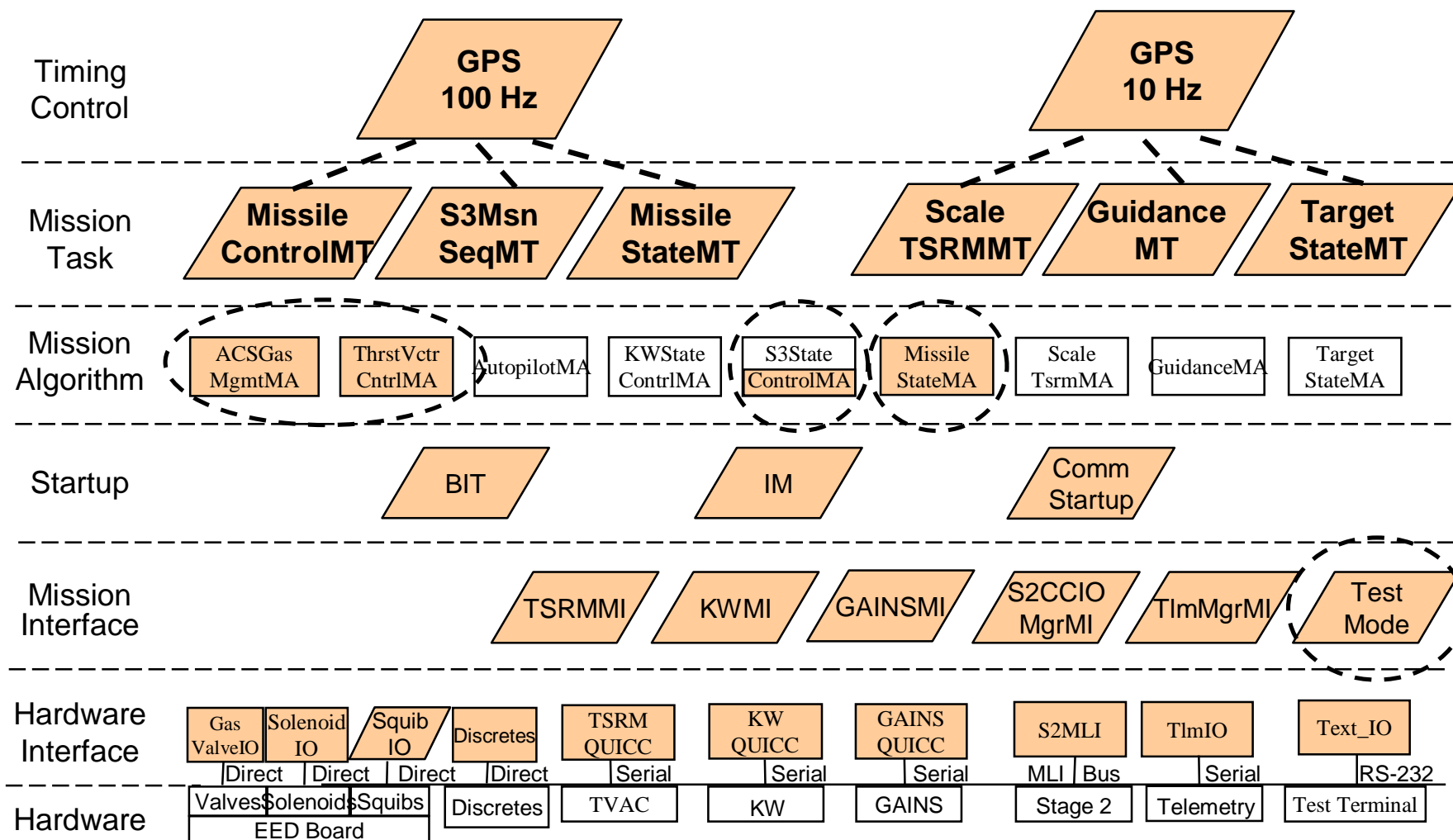


UNCLASSIFIED

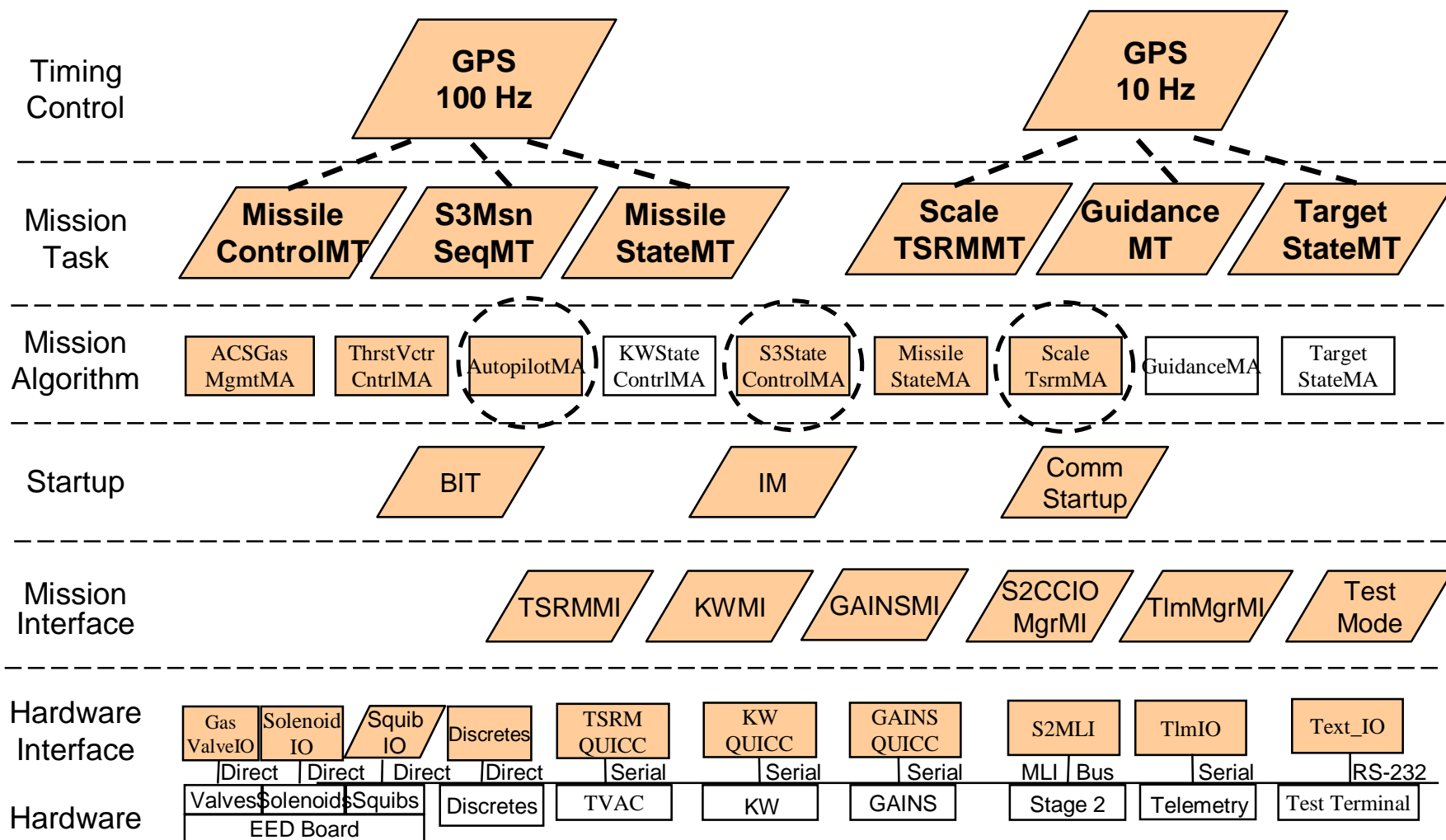
Build 1.2 - Complete Architecture Skeleton



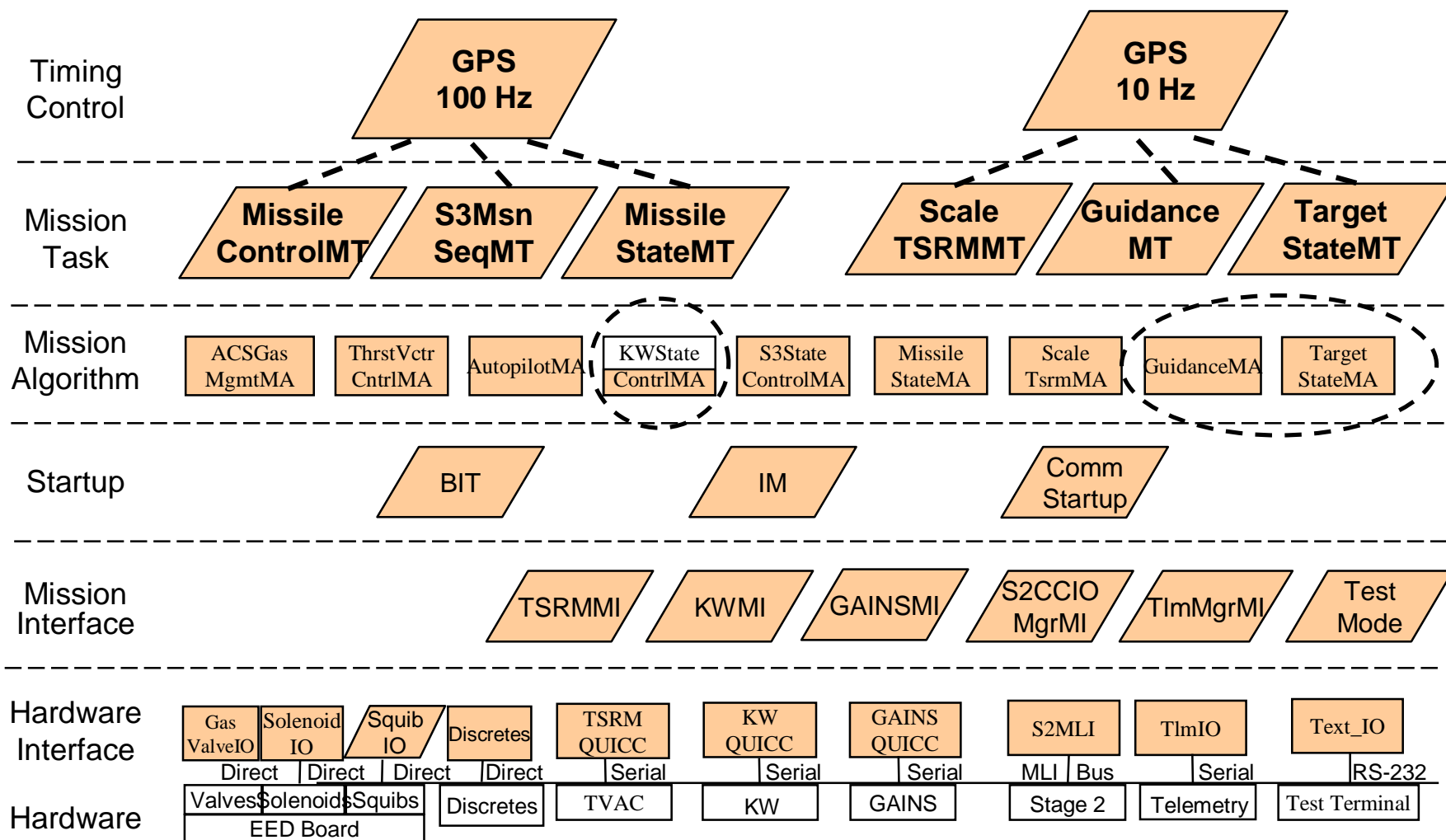
Build 1.3 - Through Pre-Separation



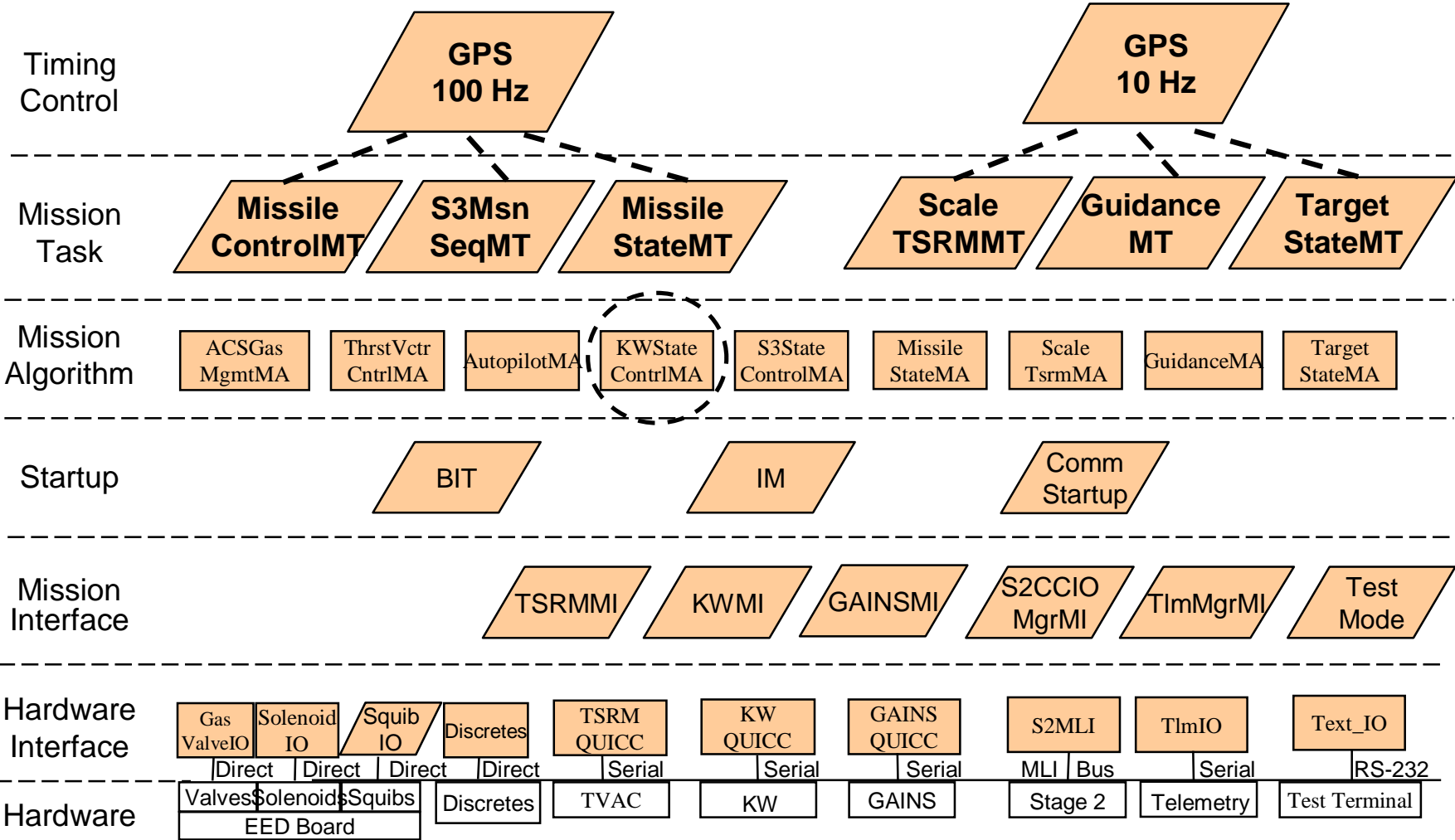
Build 1.4 - Missile Control



Build 1.5 - Missile Guidance



Build 1.6 (Build 2)- KW Preparation and Release



SM Block IV Kernel

- **Very Small, Fast Task Switch,**
- **No Ada Tasks, Block IV-Kernel Tasks**
- **Task scheduling method: fixed priority, preemptive**
 - Tasks can be preempted by a higher priority task, or an interrupt
 - Good for fast throughput, hard deadlines
- **Tasks signal each other via semaphore calls provided by Blk IV Kernel**
 - Kernel.Waitfor(Semaphore_Name);
 - Kernel.Signal(Semaphore_Name);

Task Purpose, Priorities, & Timing

Task	Priority	Timing	Task	Priority	Timing
Squib Manager	1	Event-Driven Spaces Squib Firings	TSRM	13	10 Hz
100 Hz Task	2	100 Hz, Drives other /100Hz Ttasks	Motor Scaling		References Motor Table, generates some Burnout-Reference Guidance Values
GAINS I/F	3	Event -Driven, on Arrival of GAINS Msg	Target State	14	10 Hz Maintains Current Data on the Target
Missile State	4	100 Hz Maintains Missile State Data	KW I/F	15	Event -Driven on Arrival of KW Msg
Missile Control	5	100 Hz Drives 100 Hz Autopilot Components	BIT	16	100 Hz, Collects Stage 3 BIT, forward BIT Requests and BIT Reports
TSRM I/F	6	Event -Driven on Arrival of TSRM Msg	Telemetry	17	50 Hz, Sends Tlm Buffer, Collect Some Telemetry Data
S3 Mission Sequencer	7	100 Hz Maintains Mission Timeline	Guidance	18	10 Hz, Generates the Guidance Acceleratin Vector
IM Message Processing	8	Event Driven on Arrival of IM or IM responses	Testmode	19	Event-Driven, on arrival on Testmode Command String
Build1Sequencer	9	100 Hz Drives Interface Tests	Menu Task	20	Event Driven, on Arrival of Operator Cmnds or Scripts
Stage 2 I/F	10	Event -Driven on Arrival of S2 Msg Msg	Idle Task	21	Always runs when other tasks blocked
CommStartup	11	100 Hz Drives Startup Protocols for Interfaces			
10Hz Task	12	10 Hz Drives other 10 Hz Tasks			

Reuse - Algorithms

- **Algorithm Reuse by SM3:**

- Terrier-Leap Mission Algorithms Reused in Mission Algorithm Layer
 - Autopilot, Guidance, Missile State, Target State, Scale TSRM, etc.

Reuse - Architecture Typing: “Archi-typing”

- **Suggest that this Architecture Represents an Archi-type for Embedded Missile System (EBMS) Software:**
 - The Architecture Can be Adopted for Reuse in other EBMS applications
- **A Domain Analysis Could Lead to:**
 - Better Identification of EBMS sub-domains
 - A detailed Identification of commonalties and differences between applications within an EBMS Domain
 - These commonalties/differences could be used to “genericise” the architecture for easier instantiation within a domain